NPS-53-86-0005

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A USER FRIENDLY MESH GENERATOR USING THE "C" PROGRAMMING LANGUAGE

by

Michael Hartmann

February 1986

Technical Report For Period

March 1985 - October 1985

Approved for public release; distribution unlimited

Prepared for: Naval Postgraduate School
Monterey, CA 93943

FedDocs
D 208.14/2
NPS-53-86-0005

FedDocs
D 208.14/2
NPS-53-86-0005

NAVAL POSTGRADUATE SCHOOL
MONTEREY CALIFORNIA 93943

R. H. SHUMAKER
Rear Admiral, U. S. Navy
Superintendent

D. A. SCHRADY
Provost

Reproduction of all or part of this report is authorized.

This report was prepared by:

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
NPS-53-86-0005

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A user friendly mesh generator using the "C" Programming Language

5. TYPE OF REPORT & PERIOD COVERED
Technical Report
03/85 - 10/85

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Michael Hartmann

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE
February 1986

13. NUMBER OF PAGES
44

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
finite elements, isoparametric concept, mesh generator, IBM PC, "C" programming language

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report includes the documentation and source listing of an user friendly interactive computer program that generates the element connectivity and nodal point coordiates for two dimensional isoparametric finite elements using techniques such as nested menus, keyboard filtering, dynamamic memory allocation, graphics support, preparation of screen output in RAM memory and module communication via unformatted disk files in order to take full advantage of small machines with limited RAM memory. The maximum number of finite elements is limited only by the available disk space.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

S/N 0102-LF-014-6601

A user friendly mesh generator using the C programming language

Table of contents

(1) Introduction

(2) Hardware Requirements

(3) The Mesh generator concept
- (3.1) step 1
- (3.2) step 2
- (3.3) step 3
- (3.4) step 4
- (3.5) summary

(4) Documentation of the program
- (4.1) The main menu
- (4.2) Input
- (4.3) (fine) mesh generation
- (4.4) (fine) mesh output

(5) Some remarks on the implemented software techniques
- (5.1) dynamic memory allocation
- (5.2) module communication via files
- (5.3) keyboard filtering
- (5.4) preparation of screen output in RAM memory

(6) file structure
- (6.1) '.key' files
- (6.2) '.con' files
- (6.3) '.coo' files
- (6.4) '.loc' files

(7) References

(8) Source Listings
- (8.1) meshmenu.c
- (8.2) inputcrt.c
- (8.3) keyimage.c
- (8.4) keygraf.c
- (8.5) keychnge.c
- (8.6) meshgen.c
  - (8.6.1) mshiptrd.c (called by meshgen.c)
  - (8.6.2) quad2d.c (called by meshgen.c)
- (8.7) meshimge.c
- (8.8) meshgraf.c

(1) Introduction

This software package generates the element connectivity and nodal point coordinates for two dimensional isoparametric 8 nodal point finite elements.
The maximum number of finite elements is limited only by the available disk space.
Furthermore several techniques such as

- nested menus
- keyboard filtering
- graphics support
- preparation of screen output in RAM memory
- dynamic memory allocation
- module communication via disk files

are employed in order to achieve a user friendly, fast executing system that takes full advantage of small machines with limited RAM memory.

(2) Hardware requirements

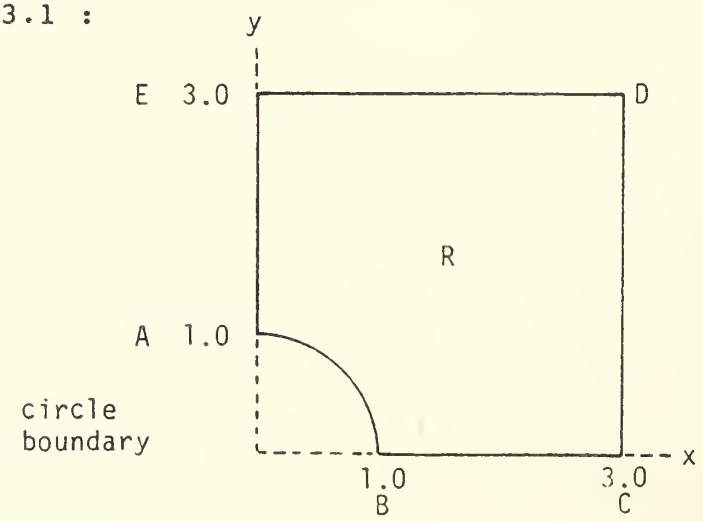
The hardware configuration necessary to run this program is an IBM PC or compatible with 256K of RAM memory and 2 DS/DD disk drives. The PC must be equipped with a video card that supports the high resolution graphics mode. Advantageous but not necessary is a hard disk.
Depending upon what floating point library is linked in with the compiled code, the program may be used with the 8087 (rsp. 80287 for the PC/AT) math coprocessor chip or an 8086/8088 (rsp. 80286 for the PC/AT) chip.

(3) The mesh generator concept

The theory of isoparametric elements is discussed in full detail in ref.[1] (pp.9-76). We will use the notation of ref.[1].

An example shall demonstratethemesh generator concept.
Let us assume we want to discretize the region shown in Fig. 3.1.

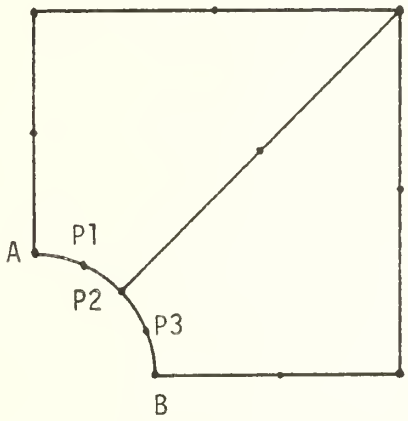
Fig. 3.1 :



(3.1) Step 1 : "Split" R into cartesian super elements (CSEL).

Each CSEL corresponds to one isoparametric quadratic 8 nodal point real element (see ref. [1],p.21/39/104).
We could choose two CSELs as indicated in Fig. 3.2.

Fig. 3.2 :

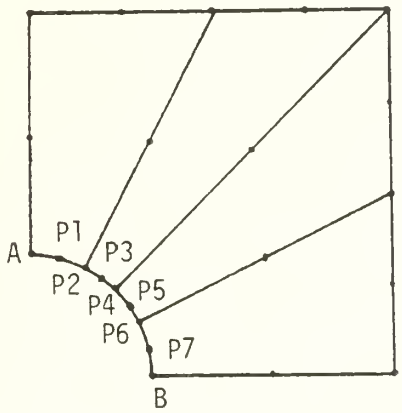


A,P1,P2,P3,B are located on the circle boundary

The circle boundary AB in Fig. 3.1 is now approximated by the two quadratic polynomials passing through (A,P1,P2) and (P2,P3,B).

We get a "better" approximation if we choose 4 CSELs .

Fig. 3.3 :



A,P1,..,P7,B are located on the circle boundary

Now the circle boundary AB is approximated by the 4 quadratic polynomials defined by (A,P1,P2),(P2,P3,P4),(P4,P5,P6) and (P6,P7,B).
It should be mentioned that in both cases (2 CSEL/Fig.3.2 and 4 CSEL/Fig.3.3) the remaining boundary BCDE of R is represented exactly by the 8nodal point elements.
Depending on how "accurate" we want to approximate the actual boundary of the given region R, we have to "split" R in adequately many CSEL's.
If R consists of different materials, we have to decompose RB with respect to the material and "split" each piece (now consisting of one material) into CSELs.

(3.2) Step 2 : Transform the CSELs into master grid super elements (MSEL).

The master grid consists - in this program - of 10x10 MSELs. Each MSEL has a unique identification number as shown in Fig. 3.4 :

Fig. 3.4 : identification numbers of the MSELs

	global col 1	col 2								col 10
globalrow 1	1	11	21	31	41	51	61	71	81	91
globalrow 2	2	12	22	32	42	52	62	72	82	92
	3	13	23	33	43	53	63	73	83	93
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
globalrow 10	10	20	30	40	50	60	70	80	90	100

Continuing with the example given in Fig. 3.1 and 3.2 we could transform the two CSELs into any two adjacent MSELs within the master grid. In principle we have only two different choices : either two MSELs in one global row (e.g. : MSEL 1 and 11) or in one global column (e.g. : MSEL 1 and 2).
It turns out that the choice MSEL 1 and 11 results in a lower bandwidth of the global linear system (in comparison with the choice MSEL 1 and 2).

Each chosen MSEL corresponds to one isoparametric quadratic 8 nodal point reference element (see ref.[1],p.21/39/104).
We call a global row rsp. column active, if it exists at least one chosen MSEL in that row rsp. column.

The reference element numbering is assumed to be as follows :

Fig. 3.5 : reference element numbering

```
              8
1  *-----*-----*  7
   |           |
2  *           *  6
   |           |
3  *-----*-----*  5
              4
```

(3.3) Step 3 : Calculate the 8 (x,y)-coordinates of each CSEL.

In the case of the region R given in Fig. 3.1 we get the following values :

Table 3.1 :

CSEL (corresponding to MSEL 1) :

	x	y	coordinate
nodal point 1 :	0.0	3.0	
2 :	0.0	2.0	
3 :	0.0	1.0	
4 :	0.3827	0.9239	
5 :	0.707	0.707	
6 :	1.8535	1.8535	
7 :	3.0	3.0	
8 :	1.5	3.0	

CSEL (corresponding to MSEL 11) :

	x	y	coordinate
nodal point 1 :	3.0	3.0	
2 :	1.8535	1.8535	
3 :	0.707	0.707	
4 :	0.9239	0.3827	
5 :	1.0	0.0	
6 :	2.0	0.0	
7 :	3.0	0.0	
8 :	3.0	1.5	

Because we deal with isoparametric elements, the information given in table 3.1 is sufficient to construct the (coarse) mesh shown in Fig. 3.2.

(3.4) Step 4 : Decide how to refine the (coarse) mesh defined by the CSELs.

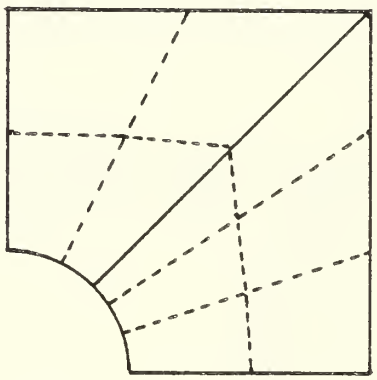
Finally we have to define the refinement of the (coarse) mesh. We do this by assigning every active global row rsp. column an positive integer value called the number of subrows rsp. subcolumns.

Let us resume the example given by Fig. 3.1 and 3.2. If we would define :

number of subrows of global row 1 : 2
number of subcols of global col 1 : 2
number of subcols of global col 2 : 3 ,

the mesh generator would produce the (fine) mesh shown in Fig. 3.6.

Fig 3.6 :



(3.5) Summary of the concept

	Enter during input :
Step 1 : Choose CSELs	
Step 2 : Transform CSELs into MSELs	identification numbers
Step 3 : Compute coordinates of CSELs	(x,y)-coordinates
Step 4 : Define refinement	nr of subrows/subcols

(4) Documentation

With the use of nested menus and keyboard filtering the program requires no learning on the user's part. The menus are self--explanatory and channel the user in the desired direction.
Input data is entered from the keyboard corresponding to the last chapter.

(4.1) The main menu

To run the program one has to load the executable file named MESHMENU.EXE into RAM memory by typing MESHMENU followed by a carriage return (compare with the source 'meshmenu.c',see 8.1).
When the progam is executed, the main menu is displayed :

Menu 4.1 :

```
 ------------------------------------------------------------------
|                                                                  |
|              This is the mesh generator main menu !              |
|                                                                  |
|                                                                  |
|   If you want to                                            HIT  |
|                                                                  |
|   -  create an mesh generator input  file                    1   |
|   - display an mesh generator input  file                    2   |
|   - display an mesh generator input  file graphically        3   |
|   -  change an mesh generator input  file                    4   |
|   - run the mesh generator (an input file must exist )       5   |
|   - display an mesh generator output file                    6   |
|   - display an mesh generator output file graphically        7   |
|   -  change an mesh generator output file                    8   |
|                                                                  |
|   -  exit to DOS                                             9   |
|                                                                  |
|                                                                  |
|   Hit a key (1-9) : @                                            |
|                                                                  |
 ------------------------------------------------------------------
```

The cursor will be located at the character position indicated by the '@' symbol.

As one can see the package consists of three major parts :

	choice
1. input phase of the (coarse) mesh	1-4
2. (fine) mesh generation	5
3. (fine) mesh output	6-7

Choice 8 is not implemented yet.
Choice 9 returns control to DOS.

(4.2) Input phase of the (coarse) mesh

Choice 1 : 'create an mesh generator input file '

The module INPUTCRT (see 8.2) is executed.
The user is asked for (compare with chapter 3)
1. a mesh generator input filename
2. the total number of CSELs
3. the identification numbers of the MSELs
4. the number of subrows rsp. subcolumns of the active global rows rsp. global columns
5. the 8 (x,y) coordinates of each CSEL

Various explanations and tables are displayed during this input process in order to inform the user and verify the data already inputted. In addition "correct prompts" are displayed to allow changes of incorrect input values.
When the input phase has finished INPUTCRT
1. creates an mesh generator input file storing all input data.
2. writes that file with the inputted filename to the default disk drive.
3. verifies step 2 and 3 by displaying messages on the screen.
4.returns controlto MESHMENU and the main menu is displayed.

Choice 2 : 'display an mesh generator input file'

The module KEYIMAGE (see 8.3) is executed.
The user is asked for the filename of the (existing) mesh generator input file.
All input data is displayed :
- chosen MSELs
- (x,y) coordinates of corresponding CSELs
- number of subrows rsp. subcolumns of active global rows rsp. global columns.

Finally control is given to MESHMENU and the main menu is diplayed.

Choice 3 : 'display an mesh generator input file graphically'

The module KEYGRAF (see 8.4) is executed.
The user is asked for the filename of the (existing) mesh generator input file.
The coarse mesh is displayed graphically .
Then control is given to MESHMENU and the main menu is displayed.

Choice 4 : 'change an mesh generator input file'

The module KEYCHNGE (see 8.5) is executed.
The user is asked for the filename of the (existing) mesh generator input file.

Then the following menu is displayed :

Menu 4.2 :

If you want to	Hit
- change coordinates	1
- change the number of subrows	2
- change the number of subcolumns	3
- Exit	4

If one chooses 1,2 or 3
- the program channels the user through the desired option.
- returns control to KEYCHANGE and menu 4.2 is displayed.

If one chooses 4
the program returns control to MESHMENU and the main menu is displayed.

(4.2) (fine) mesh generation

Choice 5 : 'run the mesh generator'

The module MESHGEN (see 8.6) is executed.
The user is asked to enter the filename of the existing mesh generator input file.
While the fine mesh (connectivity and coordinates) is generated, informations are displayed to inform the user about the progress of the mesh generation.
3 files with file extensions '.con','.coo' and '.loc'are created, storing connectivity, coordinates and some additional data of the fine mesh.
The files are written to the default disk drive.
The program verifies all steps by displaying several messages on the screen.
Finally control is returned to MESHMENU and the main menu is displayed.

(4.3) (fine) mesh output

Choice 6 : 'display an mesh generator output file'

The module MESHIMGE (see 8.7) is executed.
The user is asked for the filename of the corresponding mesh generator input (!) file.
Then connectivity and coordinates are displayed. Thereafter control is returned to MESHMENU and the main menu is displayed.

Choice 7 : 'display an mesh generator output file graphically'

The module MESHGRAF (see 8.8) is executed.
The user is asked for the filename of the corresponding mesh generator input (!) file.
Then the fine mesh is displayed graphically.
Again control is returned to MESHMENU and the main menu is displayed.

(5) Some remarks on the implemented software techniques

(5.1) dynamic memory allocation

Memory is allocated dynamically in all modules - except MESHMENU - for all variables if the corresponding storage space exceeds 20 bytes.
It is established by employing the C system library routines
- calloc() and
- free(). (see ref.[4] p.)

(5.2) module communication via files

To take full advantage of small machines with limited RAM memory, information is stored on files. The modules communicate exclusively via files.
Basically every module - except MESHMENU - consists of 3 parts :
- reading an file
- manipulating that file
- creating a (new) file .

In addition to an economical use of RAM memory,this technique allows the user to execute the modules independently. For example a user may run the input module INPUTCRT (choice 1), exit to DOS (choice 9) and switch off the computer. Because all information is savedon a file, he may continue his job at a later time without loosing any data that is already inputted.

(5.3) keyboard filtering

In order to ensure a high degree of user friendliness all keyboard input is filtered.
Integer and floating point filtering is established by employing the (Essential S) C library routines getint() and getfloat() (see ref.[5], p.4-46).
In addition the program checks the filenames inputted by the user:
- if the user wants to run the module INPUTCRT (choice 1) the program asks for an filename with file extension '.key'.If the user enters an invalid filename, the module displays a "non valid" message and asks for another name.

- if the user chooses one of the modules 2,3,4,5,6, or 7 the program asks for an (existing) filename with file extension '.key'. On entering a filename with a non valid file extension, the routine "behaves" as described above. If the user enters a filename with valid file extension, the program verifies whether that file does exist on the default disk drive. If the file does not exist, the module displays a "does not exist" message and asks for another filename.

(5.4) preparation of screen output in RAM memory

If the program is executed the user will recognize that the main menu in MESHMENU (see menu 4.1) or the messages of INPUTCRT are displayed almost instantaneously.
Menus and messages were edited, displayed once on the screen and then saved to the disk. Once they were saved, any program has access to these "menu" and "message" files.
The routine wndfrdsk() (see ref.[5],p.4-154) restoressuch a file into RAM memory.
For example the main menu (see menu 4.1) is saved in the file window.Ø (compare with (8.1)). After invoking wndfrdsk() in (8.1) the file information is restored into the integer array messØ[].
The routine rstwindo() (see ref.[5],p.4-1Ø5) displays the information - now saved in RAM memory - on the text mode screen.
The screen output information is first copied to the PC's graphics RAM memory and then displayed.

(6) File structures

(6.1) '.key' files

'.key' files are
- created in INPUTCRT
- read in KEYIMAGE, KEYGRAF, KEYCHNGE, MESHGEN
- changed in KEYCHNGE
- formatted files

structure :

```
-  total # of MSELs                    (assume k)
-  id. # of MSEL 1-k
-  # of "last" active  global row  (assume l)
-        - " -                 col  (assume m)
-  # of subrows of global row 1-l
-  # of subcols of global col 1-m
-  x-coord. of nodal point 1-8 of CSEL 1
-  y-coord. of nodal point 1-8 of CSEl 1
-  ...
-  ...
-  x-coord. of nodal point 1-8 of CSEL k
-  y-coord. of nodal point 1-8 of CSEL k
```

(6.2) '.con' files

'.con' files are
- created in MESHGEN
- read in MESHIMGE, MESHGRAF
- unformatted files

structure :

```
( comp         = computed
  f_m          = fine mesh
  l_n_p        = local nodal point
  n_p_number   = nodal point number in the fine mesh
  s_elem       = (sub)element       in the fine mesh   )
```

Bytes :

```
-                 2    maximum n_p_n                    (maxnodenr)
-                 4    total number of s_elem in f_m  (totelems)

-  4 +            2    s_elem number in f_m of  1st comp s_elem
-  4 +            4    id number of MSEL this s_elem belongs to
-  4 +            6    not yet used
-  4 +            8    not yet used
-  4 +           10    n_p_num  of l_n_p 1  of  1st comp s_elem
-  ...
-  4 +           24    n_p_num  of l_n_p 8  of  1st comp s_elem

-  ...
-  ...

-  4 + ...             s_elem number in f_m of last comp s_elem
-  4 + ...             id number of MSEL this s_elem belongs to
-  4 + ...             not yet used
-  4 + ...             not yet used
-  4 + ...             n_p_num  of l_n_p 1  of last comp s_elem
-  ...
-  4 + 24*totelems     n_p_num  of l_n_p 8  of last comp s_elem
```

(6.3) '.coo' files

'.coo' files are
- created in MESHGEN
- read in MESHIMGE, MESHGRAF
- unformatted files

structure :

Bytes :

```
-                4  x-coord. of n_p  1  in  1st comp  s_elem
-                8  y-coord. of n_p  1  in  1st comp  s_elem
-   ...
-               60  x-coord. of n_p  8  in  1st comp  s_elem
-               64  y-coord. of n_p  8  in  1st comp  s_elem

-   ...

-                   x-coord. of n_p  1  in last comp  s_elem
-                   y-coord. of n_p  1  in last comp  s_elem
-   ...
-                   x-coord. of n_p  8  in last comp  s_elem
-  totelems * 64    y-coord. of n_p  8  in last comp  s_elem
```

(6.4) '.loc' files

'.loc' files are
- created in MESHGEN
- read in MESHIMGE, MESHGRAF
- unformatted files

structure :
(loc = location)
Bytes :

```
-              2   loc of s_elem          1  in '.con'/'.coo' file
-  ...
-  totelems * 2   loc of s_elem  totelems  in '.con'/'.coo' file
```

(7) References :

[1] G.Dhatt/G.Touzot, The finite element method displayed, translated by G.Cantin, John Wiley&Sons, 1984

[2] J.Adamek, An automatic mesh generator using two and three-dimensional isoparametric finite elements, Thesis, Dept. of Mechanical Engineering, 1973

[3] B.Kernighan/D.Ritchie, The C programming language, Prentice Hall, Englewood Cliffs, 1978

[4] Microsoft Corporation, Microsoft C 3.Ø Compiler, 1984

[5] Essential Software, C Utility library, Maplewood, New Jersey, 1985

(8) Source Listings

(8.1) meshmenu.c

```
#include <stdio.h>
#define TRUE 1

main()
{
int messØ[1541],j;

        wndfrdsk("window.Ø",messØ);  /* meshmenu main display */
    do  {
        clrscrn();
        clearkbd();
        rstwindo(1,5,23,71,messØ);
        curlocat(22,24);
        getint(1,1,Ø,1,2,&j,1,1,9);
        curlocat(25,Ø);  /* cursor vanishes */
        clrscrn();

          switch(j) {

           case 1 : inputcrt();
                    break;
           case 2 : keyimage();
                    break;
           case 3 : keygraf();
                    break;
           case 4 : keychnge();
                    break;
           case 5 : meshgen();
                    break;
           case 6 : meshimge();
                    break;
           case 7 : meshgraf();
                    break;
           case 8 : printf("'meshchange.c' does not exist yet !");
                    break;
           case 9 : return;
          default : printf("You pressed a non valid key !");
                    break;
         } /* end switch */

        if ( (j!=2)&&(j!=3)&&(j!=4)&&(j!=6)&&(j!=7) ) {
                 curlocat(24,Ø);
                 Printf("Hit any key to continue ...");
                 getkey(&j);
           }

    } while (TRUE) ;

}
```

(8.2) inputcrt.c

```
#include <stdio.h>
#include <malloc.h>
static char s[15];
static char *name=s;

inputcrt()
{
FILE *input;
unsigned short *keygrid,*idnumber,*nr_rows,*nr_cols,lastcol=0,
        lastrow=0,i=0,k,row,col,nr,rowml,colml,nr_keyelements;
float *x_co_key,*y_co_key;
int *mess1,*mess2,*mess3,mess4[2],messblk1[1],messblk2[2];

mess1 = (int *)calloc(133,sizeof(int));
mess2 = (int *)calloc(396,sizeof(int));
mess3 = (int *)calloc(1584,sizeof(int));
keygrid  = (unsigned short *)calloc(101,sizeof(unsigned short));
idnumber = (unsigned short *)calloc(101,sizeof(unsigned short));
nr_rows  = (unsigned short *)calloc(11 ,sizeof(unsigned short));
nr_cols  = (unsigned short *)calloc(11 ,sizeof(unsigned short));

if ( mess1==NULL || mess2==NULL || mess3==NULL || keygrid==NULL ||
    idnumber==NULL || nr_rows==NULL || nr_cols==NULL ) {
   printf("*calloc() fails, not enough storage space !");
   return;
   }

wndfrdsk("wndblk.1",messblk1);  /* *                 messblk[1]  */
wndfrdsk("wndblk.2",messblk2);  /* **                messblk[2]  */
wndfrdsk("window.1",mess1);     /* node connec.      mess1[133]  */
wndfrdsk("window.2",mess2);     /* small keygrid     mess2[396]  */
wndfrdsk("window.3",mess3);     /* large keygrid     mess3[1584] */
wndfrdsk("window.4",mess4);     /* 2 blanks          mess4[2]    */

clrscrn();
   printf("During this input procedure the program creates an\n");
   printf("input file !\n");
   printf("\nEnter the name of the inputfile to be created\n");
   printf("( Use the file extension '.key' )  :  ");
label:   scanf ("%s",s);


while (s[i] != '\0') {
      if (s[i] != '.' ) { i=i+1;continue;}
      break;
      }
if((s[i+1] != 'k')||(s[i+2]!='e')||(s[i+3]!='y')) {
   printf("The inputfile has a non valid file extension '.key'\n")
   printf("Try again : ");
   goto label;
         }
```

```
   input = fopen(name,"w");

   clrscrn();
   rstwindo(0,0,23,65,mess3);
   curlocat(24,0);
   printf("Hit any key to continue ...");
   getkey(&i);

   do {
      clrscrn();
      printf("\nEnter the number of keyelements : ");
      getint(3,1,-1,1,2,&nr_keyelements,0,1,100);
      curlocat(24,0);
      printf("correct ? (Y,N) ");
      i = ecoyesno(24,16,1);
   } while ( i==0) ;

   fprintf(input," %u ",nr_keyelements);

x_co_key = (float *)calloc(8*nr_keyelements,sizeof(float));
y_co_key = (float *)calloc(8*nr_keyelements,sizeof(float));

if (x_co_key==NULL || y_co_key==NULL) {
   printf("*calloc() fails,not enough storage space !");
   return;
   }
   clrscrn();
   rstwindo(1,47,12,79,mess2);
   for (nr=1;nr<=nr_keyelements;++nr) {
    do {
       curlocat(15,0);
       clreos();
       curlocat(15,0);
       printf("Enter id. number of (first or) next keyelement : ");
       getint(2,1,-1,1,2,&idnumber[nr],0,1,99);
       curlocat(24,0);
       printf("correct ? (Y,N) ");
       i = ecoyesno(24,16,1);
    } while (i == 0);

       fprintf(input," %u ",idnumber[nr]);
       keygrid[idnumber[nr]] = 1;
       rstwindo(rowml+1,45,rowml+1,46,mess4);
       rstwindo(0,46+3*colml,0,47+3*colml,mess4);
       col   = idnumber[nr]/10.5 + 1;
       row   = idnumber[nr] - (col-1)*10;
       paint(row+1,46+3*col,row+1,48+3*col,23,0);
       rstwindo(row+1,45,row+1,45,messblk1);
       rstwindo(0,46+3*col,0,47+3*col,messblk2);
       colml = col;
       rowml = row;
       if ( lastrow < row ) lastrow = row;
       if ( lastcol < col ) lastcol = col;
  } /* end nr */
```

```
        curlocat(25,Ø);     /* cursor vanishes */
        rstwindo(rowml+1,45,rowml+1,46,mess4);
        rstwindo(Ø,46+3*colml,Ø,47+3*colml,mess4);
        savwindo(1,47,12,79,mess2); /* saving the blinking grid */
fprintf(input," %u ",lastrow);
fprintf(input," %u ",lastcol);

for (row=1;row<=lastrow;++row) {
    curlocat(row+1,45);
    putchar(16);
    paint(row+1,45,row+1,45,23,Ø);      /* makes it blink */

 do {
    curlocat(12+row,Ø);
    clreos();
    curlocat(12+row,Ø);
    printf("Enter number of subrows  in     globalrow %u : ",row);
    clearkbd();
    getint(2,1,-1,1,2,&nr_rows[row],Ø,1,99);
    curlocat(24,Ø);
    printf("correct ? (Y,N) ");
    i = ecoyesno(24,16,1);
  } while (i==Ø);
    fprintf(input," %u ",nr_rows[row]);
    rstwindo(row+1,45,row+1,46,mess4);
   }

for (col=1;col<=lastcol;++col) {
    curlocat(Ø,46+3*col);
    putchar('\31');
    paint(Ø,46+3*col,Ø,46+3*col,23,Ø);

 do {
    curlocat(12+col,Ø);
    clreos();
    curlocat(12+col,Ø);
    printf("Enter number of subcolumns  in globalcolumn %u : ",col);
    getint(2,1,-1,1,2,&nr_cols[col],Ø,1,99);
    curlocat(24,Ø);
    printf("correct ? (Y,N) ");
    i = ecoyesno(24,16,1);
  } while (i==Ø);

    fprintf(input," %u ",nr_cols[col]);
    rstwindo(Ø,46+3*col,Ø,47+3*col,mess4);
    }
clrscrn();
printf("The system will now ask for coordinates !\n\n\n");
printf("The node connec. within the (reference) keyelements\n");
printf("is as follows :");
rstwindo(6,13,12,31,mess1);
curlocat(24,Ø);
printf("Hit any key to continue ...");
syspause(Ø,Ø,1,Ø);
```

```
getkey(&i);
clrscrn();
rstwindo(0,60,6,78,mess1);

printf("Enter the node coordinate at actual cursorposition  \n");
printf("and hit 'ENTER'. ");
curlocat(7,0);
for (nr=1;nr<=nr_keyelements;++nr) {
    col = idnumber[nr]/10.5 + 1;
    row = idnumber[nr] - (col-1)*10;
    printf("identification number  : %3u  \n",idnumber[nr]);
    printf(" in     global column  : %3u  \n",col);
    printf("and     global    row  : %3u  \n\n",row);
    printf("         x-coordinate      y-coordinate  \n");

    for (k=1;k<=8;++k) {
    do {
       curlocat(11+k,0);
       clreos();
       rstwindo(8,47,19,79,mess2);
       rstwindo(row+8,45,row+8,45,messblk1);
       rstwindo(7,46+3*col,7,47+3*col,messblk2);
       curlocat(11+k,0);
       printf("node %lu : ",k);
       getfloat(6,6,1,-1,1,2,&x_co_key[(nr-1)*8+k],0,-1000.,1000.);
       curlocat(11+k,27);
       getfloat(6,6,1,-1,1,2,&y_co_key[(nr-1)*8+k],0,-1000.,1000.);
       curlocat(24,0);
       printf("correct ? (Y,N) ");
       i = ecoyesno(24,16,1);
       } while ( i==0 );

       } /* end k */

    for (k=1;k<=8;++k) fprintf(input," %f ",x_co_key[(nr-1)*8+k]);
    for (k=1;k<=8;++k) fprintf(input," %f ",y_co_key[(nr-1)*8+k]);
    curlocat(7,0);
    clreos();
 } /* end nr */
    fclose(input);
free(mess1);
free(mess2);
free(mess3);
free(keygrid);
free(idnumber);
free(nr_rows);
free(nr_cols);
free(x_co_key);
free(y_co_key);
clrscrn();
printf("The inputfile with name :  %s  \n",s);
printf("is created and saved on the actual discdrive.\n\n\n");
} /* end inputcrt */
```

(8.3) keyimage.c

```
#include <stdio.h>
#include <malloc.h>
static char s[15];
static char *name = s;

keyimage()
{
FILE *input;
unsigned short *keygrid,*idnumber,lastcol,lastrow,j,k,row,col,
               *nr_rows,*nr_cols,nr,nr_keyelements;
float *x_co_key,*y_co_key;
int *mess1,*mess2;

mess1 = (int *)calloc(133,sizeof(int));
mess2 = (int *)calloc(396,sizeof(int));
keygrid = (unsigned short *)calloc(101,sizeof(unsigned short));
idnumber = (unsigned short *)calloc(101,sizeof(unsigned short));
nr_rows =(unsigned short *)calloc(11,sizeof(unsigned short));
nr_cols = (unsigned short *)calloc(11,sizeof(unsigned short));


   clrscrn();
if (   mess1==NULL ||   mess2==NULL || keygrid==NULL ||
   idnumber==NULL  ||   nr_rows==NULL || nr_cols==NULL )
   {
   printf(" *calloc() fails, not enough storage space !");
   return;
   }

wndfrdsk("window.1",mess1);       /*  node connectivity  */
wndfrdsk("window.2",mess2);       /*  small keygrid      */

    printf("This subroutine displays the data of an existing   \n"
    printf("meshgenerator inputfile.\n\n");
    printf("Enter name of inputfile : ");
    scanf("%s",s);
    while ( filexist(name) == 0 ) {
       printf("\nThe file  %s  does not exist ! Try again : ",s);
       scanf("%s",s);
       }
    input = fopen(name,"r");
    fscanf(input,"%u",&nr_keyelements);

x_co_key = (float *)calloc(8*nr_keyelements,sizeof(float));
y_co_key = (float *)calloc(8*nr_keyelements,sizeof(float));

if ( x_co_key==NULL || y_co_key==NULL ) {
   printf("*calloc() fails for x_co_key/y_co_key !");
   return;
   }
```

```
    for (nr=1;nr<=nr_keyelements;++nr) {
        fscanf(input,"%u",&idnumber[nr]);
        keygrid[idnumber[nr]] = 1;
    }
    fscanf(input,"%u",&lastrow);
    fscanf(input,"%u",&lastcol);
    for (row=1;row<=lastrow;++row) fscanf(input,"%u",&nr_rows[row]);
    for (col=1;col<=lastcol;++col) fscanf(input,"%u",&nr_cols[col]);
    for (nr=1;nr<=nr_keyelements;++nr) {
        for (k=1;k<=8;++k) fscanf(input,"%f",&x_co_key[(nr-1)*8+k]);
        for (k=1;k<=8;++k) fscanf(input,"%f",&y_co_key[(nr-1)*8+k]);
        } /* end nr */
    fclose(input);


/* image of input */

clrscrn();
printf("Here the display of the\n");
printf("meshgenerator input file : \n\n\n");
printf("     %s",s);
curlocat(9,4);
printf("(active keyelements )");
curlocat(10,4);
printf("(are blinking.      )");

rstwindo(1,43,12,75,mess2);
for (nr=1;nr<=nr_keyelements;++nr) {
    col = idnumber[nr]/10.5 + 1;
    row = idnumber[nr] - (col-1)*10;
    paint(row+1,42+3*col,row+1,44+3*col,23,0);
    }

curlocat(0,64);
printf("nr of subrows ");
putchar('\31');
for (row=1;row<=lastrow;++row) {
    curlocat(row+1,77);
    printf("%2u",nr_rows[row]);
    }
curlocat(14,46);
putchar('\30');
printf("  nr of subcolumns");
curlocat(13,44);
for (col=1;col<=lastcol;++col) printf("%3u",nr_cols[col]);

for (nr=1;nr<=nr_keyelements;++nr) {
    col = idnumber[nr]/10.5 +1;
    row = idnumber[nr] - (col-1)*10;
    curlocat(15,0);
       rstwindo(16,60,22,78,mess1);
       printf("idnumber of keyelement  : %2u\n",idnumber[nr]);
       for (k=1;k<=8;++k)  printf("x%lu :  %12f   y%lu :  %12f\n",
                   k,x_co_key[(nr-1)*8+k],k,y_co_key[(nr-1)*8+k]);
```

```
        curlocat(24,50);
        printf("Hit any key to continue ..");
        getkey(&j);
        curlocat(15,0);
        clreos();
    } /* end nr */

free(mess1);
free(mess2);
free(keygrid);
free(idnumber);
free(nr_rows);
free(nr_cols);
free(x_co_key);
free(y_co_key);

} /* end keyimage */
```

(8.4) keygraf.c

```
#include <stdio.h>
#include <malloc.h>
char s[15];
char *name=s;

keygraf()
{
FILE *input;
unsigned short nr_keyelements,nr,j,k,factor_x,lastrow,lastcol,
               nr_rows,nr_cols,row,col,idnumber,factor_y,factor;
int *x_int,*y_int;
float *x,*y,*x_co_key,*y_co_key,*x_key,*y_key;
float min_x=10000.,min_y=100000.,max_x=-10000.,max_y=-10000.;

x_int = (int *)calloc(25,sizeof(int));
y_int = (int *)calloc(25,sizeof(int));
x = (float *)calloc(25,sizeof(float));
y = (float *)calloc(25,sizeof(float));
x_key = (float *)calloc(9,sizeof(float));
y_key = (float *)calloc(9,sizeof(float));

if (x_int==NULL || y_int==NULL || x==NULL || y==NULL ||
    x_key==NULL || y_key==NULL )
  {
  printf("*calloc() fails !");
  return;
  }


clrscrn();
printf("Enter the input filename to be displayed graphically : ");
scanf("%s",s);
    while ( filexist(name) == 0 ) {
       printf("\nThe file  %s  does not exist ! Try again : ",s);
       scanf("%s",s);
       }

/* read inputdatafile */
    input = fopen(name,"r");
    fscanf(input,"%u",&nr_keyelements);
    for (nr=1;nr<=nr_keyelements;++nr) fscanf(input,"%u",&idnumber);
    fscanf(input,"%u",&lastrow);
    fscanf(input,"%u",&lastcol);
    for (row=1;row<=lastrow;++row)  fscanf(input,"%u",&nr_rows);
    for (col=1;col<=lastcol;++col)  fscanf(input,"%u",&nr_cols);

j=8*nr_keyelements +1;
x_co_key =(float *)calloc(j,sizeof(float));
y_co_key =(float *)calloc(j,sizeof(float));
```

```
if (x_co_key==NULL || y_co_key==NULL )
   {
   printf("*calloc() fails !");
   return;
   }

    for (nr=1;nr<=nr_keyelements;++nr) {
        for (k=1;k<=8;++k) fscanf(input,"%f",&x_co_key[(nr-1)*8+k]);
        for (k=1;k<=8;++k) fscanf(input,"%f",&y_co_key[(nr-1)*8+k]);
        }
    fclose(input);

for (k=1;k<=8*nr_keyelements;++k) {
    if (min_x >= x_co_key[k] ) min_x = x_co_key[k];
    if (min_y >= y_co_key[k] ) min_y = y_co_key[k];
    if (max_x <= x_co_key[k] ) max_x = x_co_key[k];
    if (max_y <= y_co_key[k] ) max_y = y_co_key[k];
    }
factor_x = 303.0/(max_x-min_x);
factor_y = 183.0/(max_y-min_y);
if ( factor_x >= factor_y ) factor = factor_y;
else                        factor = factor_x;

grafinit(2,0,0);

for (nr=1;nr<=nr_keyelements;++nr) {
    for (j=1;j<=8;++j) {
        x_key[j] = x_co_key[(nr-1)*8+j];
        y_key[j] = y_co_key[(nr-1)*8+j];
        }
    for (j=1;j<=8;++j) {
        x[1+(j-1)*3] = x_key[j];
        y[1+(j-1)*3] = y_key[j];
        }
    for (j=1;j<=2;++j) {
     quad_2d(-1.0       ,1.0 -j/3.0,x_key,y_key,&x[j+ 1],&y[j+ 1]);
     quad_2d(-1.0       ,  -(j/3.0),x_key,y_key,&x[j+ 4],&y[j+ 4]);
     quad_2d(-1.0+j/3.0,-1.0       ,x_key,y_key,&x[j+ 7],&y[j+ 7]);
     quad_2d(      j/3.0,-1.0       ,x_key,y_key,&x[j+10],&y[j+10]);
     quad_2d( 1.0       ,-1.0+j/3.0,x_key,y_key,&x[j+13],&y[j+13]);
     quad_2d( 1.0       ,      j/3.0,x_key,y_key,&x[j+16],&y[j+16]);
     quad_2d( 1.0-j/3.0, 1.0       ,x_key,y_key,&x[j+19],&y[j+19]);
     quad_2d(  -(j/3.0), 1.0       ,x_key,y_key,&x[j+22],&y[j+22]);
        }
    for (j=0;j<=23;++j) {
        x_int[j] = 8   +  factor * ( x[j+1] -  min_x );
        y_int[j] = 191 -  factor * ( y[j+1] -  min_y );
        }
     grpoly(x_int,y_int,24,1);
   }  /* end nr */
grxlab("Hit any key to continue..",7,55,0);
getkey(&j);
grafinit(0,0,0);
```

```
free(x_int);
free(y_int);
free(x);
free(y);
free(x_key);
free(y_key);
free(x_co_key);
free(y_co_key);

}
```

(8.5) keychnge.c

```
#include <stdio.h>
#include <malloc.h>
static char s[15];
static char *name = s;

keychnge()
{
FILE *input;
unsigned short *keygrid,*idnumber,lastcol,lastrow,j,k,row,col,
                *nr_rows,*nr_cols,idnr,nr,nr_keyelements;
float *x_co_key,*y_co_key;
int i,*mess1,*mess2;

mess1 = (int *)calloc(133,sizeof(int));
mess2 = (int *)calloc(396,sizeof(int));
keygrid = (unsigned short *)calloc(101,sizeof(unsigned short));
idnumber = (unsigned short *)calloc(101,sizeof(unsigned short));
nr_rows =(unsigned short *)calloc(11,sizeof(unsigned short));
nr_cols = (unsigned short *)calloc(11,sizeof(unsigned short));


   clrscrn();
if (   mess1==NULL ||   mess2==NULL || keygrid==NULL ||
       idnumber==NULL  || nr_rows==NULL || nr_cols==NULL )
   {
   printf(" *calloc() fails, not enough storage space !");
   return;
   }

wndfrdsk("window.1",mess1);       /*  node connectivity     */
wndfrdsk("window.2",mess2);       /*  small keygrid         */

    printf("This subroutine allows you to change an existing\n");
    printf("meshgenerator inputfile.\n\n");
    printf("Enter name of inputfile : ");
    scanf("%s",s);
    while ( filexist(name) == 0 ) {
       printf("\nThe file  %s  does not exist ! Try again : ",s);
       scanf("%s",s);
       }
    input = fopen(name,"r");
    fscanf(input,"%u",&nr_keyelements);

x_co_key = (float *)calloc(8*nr_keyelements+1,sizeof(float));
y_co_key = (float *)calloc(8*nr_keyelements+1,sizeof(float));

if ( x_co_key==NULL || y_co_key==NULL ) {
   printf("*calloc() fails for x_co_key/y_co_key !");
   return;
   }
```

```
    for (nr=1;nr<=nr_keyelements;++nr) {
        fscanf(input,"%u",&idnumber[nr]);
        keygrid[idnumber[nr]] = 1;
    }
    fscanf(input,"%u",&lastrow);
    fscanf(input,"%u",&lastcol);
    for (row=1;row<=lastrow;++row) fscanf(input,"%u",&nr_rows[row]);
    for (col=1;col<=lastcol;++col) fscanf(input,"%u",&nr_cols[col]);
    for (nr=1;nr<=nr_keyelements;++nr) {
        for (k=1;k<=8;++k) fscanf(input,"%f",&x_co_key[(nr-1)*8+k]);
        for (k=1;k<=8;++k) fscanf(input,"%f",&y_co_key[(nr-1)*8+k]);
        } /* end nr */
    fclose(input);


do {
       clrscrn();
       printf(" If you want to                                HIT \n\n");
       printf(" change  coordinates                            1  \n");
       printf(" change the number of subrows                   2  \n");
       printf(" change the number of subcolumns                3   \n");
       printf(" Exit                                           4   \n");
       printf("\nHit a key (1-4) : ");
       curlocat(7,18);
       getint(1,1,0,1,2,&j,1,1,8);
       clrscrn();

     switch(j)  {

      case 1 :   clrscrn();
                 rstwindo(1,47,12,79,mess2);
                 for (nr=1;nr<=nr_keyelements;++nr)
                    { /* keygrid blinks */
                    col = idnumber[nr]/10.5 + 1;
                    row = idnumber[nr] - (col-1)*10;
                    paint(row+1,46+3*col,row+1,48+3*col,23,0);
                    }
                 do {
                    curlocat(14,0);
                    clreos();
                    curlocat(14,0);
                    printf("Enter id# of  element  to  be  changed :");
                    curlocat(14,41);
                    getint(2,1,-1,1,2,&idnr,0,1,99);
                    curlocat(24,0);
                    printf("correct ? (Y,N) ");
                    i = ecoyesno(24,16,1);
                 } while (i == 0);

                for (nr=1;nr<=nr_keyelements;++nr)
                    if (idnumber[nr]==idnr) break;
                clrscrn();
printf("                old                                  new  ");
printf("    correct  \n\n");
                for (k=1;k<=8;++k)
```

```
      printf("x%lu : %10f  y%lu : %10f\n",k,x_co_key[(nr-1)*8+k]
                                          ,k,y_co_key[(nr-1)*8+k,k]);
for (k=1;k<=8;++k) {
do {
   curlocat(k+1,69);
   printf("%lu(Y/N)? : ",k);
   i=ecoyesno(k+1,79,1);
   if (i==0) {
   curlocat(k+1,38);
   clreol();
   curlocat(22,0);
   printf("Enter new coord. at cursorposition\nand hit 'ENTER'");
   curlocat(k+1,34);
   printf("x%lu : ",k);
   curlocat(k+1,39);
   getfloat(6,6,1,-1,1,2,&x_co_key[(nr-1)*8+k],0,-1000.,1000.);
   curlocat(k+1,54);
   printf("y%lu : ",k);
   curlocat(k+1,59);
   getfloat(6,6,1,-1,1,2,&y_co_key[(nr-1)*8+k],0,-1000.,1000.);
   curlocat(22,0);
   clreos();
   } /* end if */
   } while(i==0);
   } /* end k */
   break;


     case 2 :
   clrscrn();
   printf("                              old       new    ");
   printf("        correct\n\n");
   for (row=1;row<=lastrow;++row)
   printf("nr of subrows in row %lu : %5u\n",row,nr_rows[row]);
   for (row=1;row<=lastrow;++row) {
   do {
   curlocat(row+1,54);
   printf("(Y/N)? : ");
   i=ecoyesno(row+1,63,1);
   if (i==0) {
   curlocat(row+1,36);
   clreol();
   curlocat(22,0);
   printf("Enter new 'nr of subrows' at actual cursorposition");
   printf("\nand hit 'ENTER'");
   curlocat(row+1,36);
   getint(2,1,-1,1,2,&nr_rows[row],0,1,99);
   curlocat(22,0);
   clreos();
   } /* end if */
   } while(i==0);
   } /* end row */
   break;
```

```
      case 3 :
    clrscrn();
    printf("                                old       new    ");
    printf("              correct\n\n");
    for (col=1;col<=lastcol;++col)
    printf("nr of subcols in col %1u : %5u\n",col,nr_cols[col]);
    for (col=1;col<=lastcol;++col) {
                    do {
                       curlocat(col+1,54);
                       printf("(Y/N)? : ");
                       i=ecoyesno(col+1,63,1);
                       if (i==0) {
                          curlocat(col+1,36);
                          clreol();
                          curlocat(22,0);
    printf("Enter new 'nr of subcols' at actual cursorposition");
    printf("   \nand hit 'ENTER'");
                          curlocat(col+1,36);
                     getint(2,1,-1,1,2,&nr_cols[col],0,1,99);
                          curlocat(22,0);
                          clreos();
                     }
                    } while(i==0);
                   }
                  break;
        case 4 : return;
       default : break;
      }  /* end switch */
input = fopen(name,"w");
fprintf(input," %u ",nr_keyelements);
for (nr=1;nr<=nr_keyelements;++nr)
    fprintf(input," %u ",idnumber[nr]);
fprintf(input," %u ",lastrow);
fprintf(input," %u ",lastcol);
for (row=1;row<=lastrow;++row)
   fprintf(input," %u ",nr_rows[row]);
for (col=1;col<=lastcol;++col)
    fprintf(input," %u ",nr_cols[col]);
for (nr=1;nr<=nr_keyelements;++nr) {
    for (k=1;k<=8;++k)
       fprintf(input," %f ",x_co_key[(nr-1)*8+k]);
    for (k=1;k<=8;++k)
       fprintf(input," %f ",y_co_key[(nr-1)*8+k]);
    }
fclose(input);
} while (1);  /* global do */
free(mess1);
free(mess2);
free(keygrid);
free(idnumber);
free(nr_rows);
free(nr_cols);
free(x_co_key);
free(y_co_key);
} /* end keychnge */
```

(8.6) meshgen.c

```
#include <stdio.h>
#include <malloc.h>
static char s_loc[15],s_conn[15],s_coord[15];
static char *nameloc=s_loc,*nameconn = s_conn,*namecoord=s_coord;
static float *coord;
static unsigned short *loc,*connec;

meshgen()
{
FILE *locfile,*connfile,*coordfile;
unsigned short lastrow,lastcol,nr_of_subelements[11],nr_rows[11],
               nr_cols[11],*keygrid,totelems,maxnodenrs=0;
unsigned short  glcol_firstnode=1,glcol_subelement=1,glcol,glrow,
                key_firstnode,key_subelement,firstnode,subelement,
                subcol_firstnode,subcol_subelement,subcol,subrow,
                elemjump,key_coljump1,key_coljump2,coljump1,
                coljump2,memory1,memory2,help1,help2,
                noconnect,noconn1,noconn2,row,col,j,k,
                nr_of_keyelement=0,gl_nodenr_of_loc_node[9],
                count1=1,count2=0,dummy=0,idnumber;
float           *x_co_key,*y_co_key,x[9],y[9],ksi[4],eta[4],
                x_co_keyelem[9],y_co_keyelem[9];

connec  = (unsigned short *)calloc(1201,sizeof(unsigned short));
  coord = (float *)calloc(1601,sizeof(float));
keygrid  = (unsigned short *)calloc(101,sizeof(unsigned short));
x_co_key          = (float *)calloc(400,sizeof(float));
y_co_key          = (float *)calloc(400,sizeof(float));


meshinputread(s_loc,s_conn,s_coord,x_co_key,y_co_key,keygrid,
  nr_rows,nr_cols,&totelems,nr_of_subelements,&lastrow,&lastcol);

loc =(unsigned short *)calloc(totelems+1,sizeof(unsigned short));

if ( connec==NULL || coord==NULL || loc==NULL ||
     keygrid==NULL || x_co_key==NULL || y_co_key==NULL )
    {
    printf("*calloc() fails in meshgen !");
    return;
    }


connfile = fopen(nameconn,"w+b");
coordfile = fopen(namecoord,"w+b");
fwrite((unsigned short *)&maxnodenrs,sizeof(unsigned short),1,connf
fwrite((unsigned short *)&totelems  ,sizeof(unsigned short),1,connf

/* Begin of actual computation */

for (glcol=1;glcol<=lastcol;++glcol)  {
    elemjump  = 0;
    coljump1  = 0;
```

```
coljump2  = 0;
noconnect = 1;
for (row=1;row<=lastrow;++row)
    if (keygrid[(glcol-1)*10+row]==1)  {
       elemjump  = elemjump +   nr_rows[row];
       coljump1  = coljump1 + 2*nr_rows[row] + noconnect;
       coljump2  = coljump2 +   nr_rows[row] + noconnect;
       noconnect = 0;
    }
    else noconnect = 1;
key_firstnode  = glcol_firstnode;
key_subelement = glcol_subelement;
key_coljump1   = coljump1;
key_coljump2   = coljump2;

for (glrow=1;glrow<=lastrow;++glrow)  {
idnumber = (glcol-1)*10 + glrow;
if  (keygrid[idnumber] == 1)  {
    nr_of_keyelement     = nr_of_keyelement +1;
    subcol_firstnode  = key_firstnode ;
    subcol_subelement = key_subelement;

    help1   = 0;
    help2   = 0;
    noconn1 = 1;
    noconn2 = 1;
    for (row=1;row<glrow;++row) {
        j = (glcol-1)*10 + row;
        if ( keygrid[j]==1 ) {
           help1   = help1 +   nr_rows[row] + noconn1;
           noconn1 = 0;
        }
        else noconn1 = 1;
        if ( keygrid[j]==1 || keygrid[j+10]==1 ) {
           help2   = help2 + 2*nr_rows[row] + noconn2;
           noconn2 = 0;
        }
        else noconn2 = 1;
    }
    for (row=glrow;row<=lastrow;++row) {
        j = (glcol-1)*10 + row;
        if ( keygrid[j]==1 || keygrid[j-10]==1 ) {
           help1   = help1 + 2*nr_rows[row] + noconn1;
           noconn1 = 0;
        }
        else noconn1 = 1;
        if ( keygrid[j]==1 ) {
           help2   = help2 +   nr_rows[row] + noconn2;
           noconn2 = 0;
        }
        else noconn2 = 1;
    }
    for (subcol=1;subcol<=nr_cols[glcol];++subcol)  {
        firstnode  = subcol_firstnode;
        subelement = subcol_subelement;
```

```
          coljump1    = key_coljump1;
          coljump2    = key_coljump2;

          if ( subcol == 1 )                    coljump1 = help1;
          if ( subcol == nr_cols[glcol] )  coljump2 = help2;

          for (subrow=1;subrow<=nr_rows[glrow];++subrow)  {
              curget(&row,&col);
              curlocat(5,0);
              printf(" glcol   glrow   subcol   subrow ");
              curlocat(6,0);
              printf("%4u%8u%8u%9u",glcol,glrow,subcol,subrow);
              curlocat(row,col);
gl_nodenr_of_loc_node[1] = firstnode    ;
gl_nodenr_of_loc_node[2] = firstnode + 1;
gl_nodenr_of_loc_node[3] = firstnode + 2;
gl_nodenr_of_loc_node[4] = firstnode + coljump1 + 1;
gl_nodenr_of_loc_node[5] = firstnode + coljump1 + coljump2 + 2;
gl_nodenr_of_loc_node[6] = firstnode + coljump1 + coljump2 + 1;
gl_nodenr_of_loc_node[7] = firstnode + coljump1 + coljump2;
gl_nodenr_of_loc_node[8] = firstnode + coljump1;
              memory2     = gl_nodenr_of_loc_node[4];

           if( gl_nodenr_of_loc_node[5] >= maxnodenrs )
              maxnodenrs = gl_nodenr_of_loc_node[5];



  /* computation of the coordinates */
              for (k=1;k<=8;++k) {
        x_co_keyelem[k] = x_co_key[(nr_of_keyelement-1)*8+k];
        y_co_keyelem[k] = y_co_key[(nr_of_keyelement-1)*8+k];
                }

              ksi[3] = -1.0 + (2.0*subcol)/nr_cols[glcol];
              ksi[1] = ksi[3] - 2.0/nr_cols[glcol];
              ksi[2] = 0.5 * (ksi[1]+ksi[3]);
              eta[3] =  1.0 - (2.0*subrow)/nr_rows[glrow];
              eta[1] = eta[3] + 2.0/nr_rows[glrow];
              eta[2] = 0.5 * (eta[1]+eta[3]);
    for (k=1;k<=3;++k)
    quad_2d(ksi[1],eta[k],x_co_keyelem,y_co_keyelem,&x[k],&y[k]);
    quad_2d(ksi[2],eta[3],x_co_keyelem,y_co_keyelem,&x[4],&y[4]);
    for (k=3;k>=1;--k)
    quad_2d(ksi[3],eta[k],x_co_keyelem,y_co_keyelem,&x[8-k],&y[8-
    quad_2d(ksi[2],eta[1],x_co_keyelem,y_co_keyelem,&x[8],&y[8]);

      loc[subelement]           = count1 + 100*count2;
      connec[(count1-1)*12 +1] = subelement;
      connec[(count1-1)*12 +2] = idnumber;
      connec[(count1-1)*12 +3] = dummy;
      connec[(count1-1)*12 +4] = dummy;
      for (k=1;k<=8;++k) {
          connec[(count1-1)*12 +4 +k] = gl_nodenr_of_loc_node[k]
          coord[(count1-1)*16+2*k-1] = x[k];
```

```
            coord[(count1-1)*16+2*k]   = y[k];
            }

    if ((count1==100)||(subelement==totelems))  {
    fwrite((unsigned short *)connec+1,sizeof(unsigned short),
                                          12*count1,connfile);
    fwrite((float *)coord+1,sizeof(float),16*count1,coordfile);
    count1 = 0;
    count2 = count2 + 1;
    }
   count1 = count1 + 1;

                coljump1   = coljump1   - 1;
                coljump2   = coljump2   + 1;
                firstnode  = firstnode  + 2;
                subelement = subelement + 1;
            } /* end subrow */

      subcol_firstnode  = subcol_firstnode  + coljump1 + coljump2;
      subcol_subelement = subcol_subelement + elemjump;
         }     /* end subcol */

     key_subelement = key_subelement +   nr_rows[glrow];
     key_coljump1   = key_coljump1   -   nr_rows[glrow];
     key_coljump2   = key_coljump2   +   nr_rows[glrow];

     key_firstnode  = key_firstnode  + 2*nr_rows[glrow];
     noconn1        = 0;
     memory1        = 0;
     for (row=glrow+1;row<=lastrow;++row) {
         if (keygrid[(glcol-1)*10+row] == 1 ) break;
         if (                      glcol == 1 ) {
            memory1        = 1;
            key_firstnode = key_firstnode + 1;
            break;
            }
         if (keygrid[(glcol-2)*10+row] == 1 ) {
            memory1        = 1;
      key_firstnode = key_firstnode + 2*nr_rows[row] + noconn1;
            noconn1        = 0;
            }
         else noconn1 = 1;
     } /* end row */
   if (memory1 == 0) key_firstnode = key_firstnode + noconn1;

     } /* end if there is a keyelement in this glrow in this glcol */
     } /* end keyelement */
glcol_subelement = glcol_subelement + nr_of_subelements[glcol];
help1     = 0;
noconnect = 1;
memory1   = 1;
row       = 1;
while (    keygrid[ glcol   *10 + row] == 0 ) {
      if ( keygrid[(glcol-1)*10 + row] == 1 ) {
      memory1    = 0;
```

```
      helpl      = helpl + 2*nr_rows[row] + noconnect;
      noconnect = Ø;
      }   /* end if */
      else noconnect = 1;
      row = row + 1;
} /* end while */
if (memoryl==Ø) glcol_firstnode = memory2 + helpl + noconnect;
if (memoryl==1) glcol_firstnode = memory2 + 1;
}  /* end globalcolumn */

fseek(connfile,ØL,Ø);
fwrite((unsigned short *)&maxnodenrs,sizeof(unsigned short),1,
                                                        connfile);
fclose(connfile);
fclose(coordfile);
locfile=fopen(nameloc,"w+b");
fwrite((unsigned short *)loc+1,sizeof(unsigned short),totelems,
                                                         locfile);
fclose(locfile);



curlocat(9,Ø);
printf("The files    %s        \n",s_conn);
printf("      and    %s        \n",s_coord);
printf("      and    %s        \n",s_loc);
printf("are created and saved on your actual discdrive ! \n\n");
curlocat(25,Ø);

free(connec);
free(coord);
free(loc);
free(keygrid);
free(x_co_key);
free(y_co_key);



} /* end meshgen */
```

(8.6.1) mshiptrd.c

```
#include <stdio.h>
#include <malloc.h>
static char s[15];
static char *name = s;

meshinputread(s_loc,s_conn,s_coord,x_co_key,y_co_key,keygrid,
         nr_rows,nr_cols,totelems,nrofsubelem,lastrow,lastcol)
char           s_loc[],s_conn[],s_coord[];
float          x_co_key[],y_co_key[];
unsigned short keygrid[],nr_rows[],nr_cols[],*totelems,nrofsubelem[]
               ,*lastrow,*lastcol;
{
FILE *input;
unsigned short i,j,k,lastrowl,row,col,nr_keyelements,nr,
               idnumber,maxidnumber=1;
int *mess5;

mess5 =(int *)calloc(540,sizeof(int));
if ( mess5==NULL )
   {
   printf("*calloc() fails for mess5 in mshiptrd.c !");
   return;
   }

wndfrdsk("window.5",mess5);
rstwindo(5,18,16,62,mess5);
curlocat(16,57);
    do {
        scanf("%s",s);
        i=0;
        while (s[i] != '\0') {
              if (s[i] != '.' ) { i=i+1;continue;}
              break;
              }
         if((s[i+1]!='k')||(s[i+2]!='e')||(s[i+3]!='y')||
                                        (filexist(name)==0))
            {
      printf("\nThe file %s does not exist or has wrong ");
      printf(" file extension!\n",s);
      printf("Try again : ");
            }
         else break;
        } while (1);

for (j=0;j<=i;++j) {
    s_loc[j]   = s[j];
    s_conn[j]  = s[j];
    s_coord[j] = s[j];
    }
    s_loc[i+1]  = 'l';
    s_loc[i+2]  = 'o';
    s_loc[i+3]  = 'c';
    s_loc[i+4]  ='\0';
```

```
    s_conn[i+1] = 'c';
    s_conn[i+2] = 'o';
    s_conn[i+3] = 'n';
    s_conn[i+4] ='\Ø';
   s_coord[i+1] = 'c';
   s_coord[i+2] = 'o';
   s_coord[i+3] = 'o';
   s_coord[i+4] ='\Ø';
/* read inputdatafile */

    input = fopen(name,"r");
    fscanf(input,"%u",&nr_keyelements);
if (8*nr_keyelements >= 399)
   {
   printf("space for x_co_key and y_co_key too small !");
   return;
   }
    for (nr=1;nr<=nr_keyelements;++nr) {
        fscanf(input,"%u",&idnumber);
        keygrid[idnumber] = 1;
        if ( idnumber >= maxidnumber ) maxidnumber = idnumber;
    }
    fscanf(input,"%u",lastrow);
    fscanf(input,"%u",lastcol);
  lastrowl = maxidnumber -  (*lastcol-1)*1Ø;
                                    /* necessary for later displayay*/
  for (row=1;row<=(*lastrow);++row) fscanf(input,"%u",&nr_rows[row])
  for (col=1;col<=(*lastcol);++col) fscanf(input,"%u",&nr_cols[col])
    for (nr=1;nr<=nr_keyelements;++nr) {
        for (k=1;k<=8;++k) fscanf(input,"%f",&x_co_key[(nr-1)*8+k]);
        for (k=1;k<=8;++k) fscanf(input,"%f",&y_co_key[(nr-1)*8+k]);
        } /* end nr */
    fclose(input);
/* compute number of subelements[] */
for (col=1;col<=(*lastcol);++col) {
    nrofsubelem[col] = Ø;
    for (row=1;row<=(*lastrow);++row) {
        j = (col-1)*1Ø +row;
        if ( keygrid[j] ==1 )
  nrofsubelem[col] = nrofsubelem[col] + nr_rows[row]*nr_cols[col];
    } /* end row */
} /* end col */
*totelems = Ø;
/* compute toal number of (sub-) elements */
for (col=1;col<=(*lastcol);++col)
 *totelems = *totelems + nrofsubelem[col];
clrscrn();
printf("The computations will be finished when          \n\n");
printf(" glcol   glrow   subcol   subrow                  \n");
printf("%4u%8u%8u%9u",*lastcol,lastrowl,nr_cols[*lastcol],
                                          nr_rows[lastrowl]);
curlocat(25,Ø);
free(mess5);
}
```

(8.6.2) quad2d.c

```
quad_2d(ksi,eta,x,y,x0,y0)
float ksi,eta,x[],y[],*x0,*y0;
{
float n[9];
unsigned short function_nr;
*x0  = 0;
*y0  = 0;
n[1] = 0.25 * (1-ksi)     * (1+eta) * ( eta-ksi-1);
n[2] = 0.50 * (1-eta*eta) * (1-ksi);
n[3] = 0.25 * (1-ksi)     * (1-eta) * (-eta-ksi-1);
n[4] = 0.50 * (1-ksi*ksi) * (1-eta);
n[5] = 0.25 * (1+ksi)     * (1-eta) * (-eta+ksi-1);
n[6] = 0.50 * (1-eta*eta) * (1+ksi);
n[7] = 0.25 * (1+ksi)     * (1+eta) * ( eta+ksi-1);
n[8] = 0.50 * (1-ksi*ksi) * (1+eta);
for (function_nr=1;function_nr<=8;++function_nr) {
    *x0 = *x0 + x[function_nr]*n[function_nr];
    *y0 = *y0 + y[function_nr]*n[function_nr];
    }
}     /* end quad_2d */
```

(8.7) meshimge.c

```
#include <stdio.h>
#include <malloc.h>
static char s[15],s_loc[15],s_conn[15],s_coord[15];
static char *name = s,*nameloc=s_loc,*nameconn = s_conn,
            *namecoord = s_coord;
static unsigned short *loc,*connec;
static float *coord;

meshimge()
{
FILE *locfile,*connfile,*coordfile;
unsigned  j,k,count1=0,count2=0,nrsubelem,nodenrs,totelems;
int i,exitflag=1;
long help;

connec = (unsigned short *)calloc(1201,sizeof(unsigned short));
coord  = (float *)calloc(1601,sizeof(float));



    clrscrn();
    printf("This subroutine displays     connectivity     \n");
    printf("                    and     coordinates      \n");
    printf("after running the meshgenerator .           \n\n");
    printf("Enter the name of the meshgenerator input filename\n");
    printf("with file extension '.key' : ");
labell :     scanf("%s",s);
      i=0;
while (s[i] != '\0') {
      if (s[i] != '.' ) { i=i+1;continue;}
      break;
      }
if((s[i+1] != 'k')||(s[i+2]!='e')||(s[i+3]!='y')) {
   printf("The inputfile has a non valid file extension\n");
   printf("Try again : ");
   goto labell;
          }

for (j=0;j<=i;++j) {
    s_conn[j]  = s[j];
    s_coord[j] = s[j];
    s_loc[j]   = s[j];
    }
    s_conn[i+1] = 'c';
    s_conn[i+2] = 'o';
    s_conn[i+3] = 'n';
    s_conn[i+4] ='\0';
   s_coord[i+1] = 'c';
   s_coord[i+2] = 'o';
   s_coord[i+3] = 'o';
   s_coord[i+4] ='\0';
     s_loc[i+1] = 'l';
     s_loc[i+2] = 'o';
```

```
     s_loc[i+3] = 'c';
     s_loc[i+4] ='\0';

if(filexist(nameloc)==0 || filexist(nameconn)==0 ||
                           filexist(namecoord)==0) {
   printf("\n At least on of the files   %s        \n",s_conn);
   printf("                      and   %s          \n",s_coord);
   printf("                      and   %s          \n",s_loc);
   printf("does not exist !                         \n");
   printf("Try again : ");
   goto label1;
   }

locfile = fopen(nameloc,"r+b");
connfile = fopen(nameconn,"r+b");
coordfile = fopen(namecoord,"r+b");
fread((unsigned short *)&nodenrs,sizeof(unsigned short),1,
                                                  connfile);
fread((unsigned short *)&totelems,sizeof(unsigned short),1,
                                                  connfile);
loc =(unsigned short *)calloc(totelems+1,sizeof(unsigned short));
if ( loc==NULL || connec==NULL || coord==NULL ) {
   printf("*calloc() fails in meshimge !");
   return;
   }

fread((unsigned short *)loc+1,sizeof(unsigned short),totelems,
                                                     locfile);
for (k=1;k<=totelems;++k) printf("loc[%u] : %u \n",k,loc[k]);
getkey(&k);
clrscrn();
printf(" connectivity image of %s \n\n",s);
printf("nr of      | belongs to |  global nodenumber");
printf(" of local node \n");
printf("subelememt | keyelement |    ");
printf(" 1    2    3    4    5    6    7    8\n");
printf("\n");
while((totelems-count1*100) >= 100) {
     fread((unsigned short *)connec+1,sizeof(unsigned short),
                                              1200,connfile);

for (j=1;j<=100;++j) {
    k = (j-1)*12;
    printf("%6u%12u%12u%5u%5u%5u%5u%5u%5u%5u\n",connec[k+1],
    connec[k+2],connec[k+5],connec[k+6],connec[k+7],connec[k+8],
    connec[k+9],connec[k+10],connec[k+11],connec[k+12]);
    }
count1 = count1 +1;
} /* end while */

if((totelems-count1*100) != 0) {
     fread((unsigned short *)connec+1,sizeof(unsigned short),
              12*(totelems-100*count1),connfile);

for (j=1;j<=(totelems-100*count1);++j) {
```

```
    k = (j-1)*12;
    printf("%6u%12u%12u%5u%5u%5u%5u%5u%5u%5u\n",connec[k+1],
    connec[k+2],connec[k+5],connec[k+6],connec[k+7],connec[k+8],
    connec[k+9],connec[k+10],connec[k+11],connec[k+12]);
    }
} /* end if */
printf("Hit any key to continue ... ");
getkey(&j);

/* output of coordinates */
do {
clrscrn();
printf("If you want to see the coordinates of                HIT\n");
printf("        all subelements                               1 \n");
printf("        a specific subelement                         2 \n\n")
printf("   Exit to mesh mainmenu                              3  \n");
printf("\n Hit (1-3) : ");
getint(1,1,-1,1,2,&i,0,1,2);
clrscrn();

   switch (i) {

case 1 :   count1 = 0;
     while((totelems-count1*100) >= 100) {
     fread((float *)coord+1,sizeof(float),1600,coordfile);

     for (j=1;j<=100;++j) {
     help = ((count1*100+(j-1)*12)+2L)*2L;
     fseek(connfile,help,0);
     fread((unsigned short *)&nrsubelem,sizeof(unsigned short),1,
                                                        connfile);
     printf("subelem : %4u \n",nrsubelem);
     for (k=1;k<=8;++k)
     printf("                x%1u : %10f y%1u : %10f \n",k,
     coord[(j-1)*16+2*k-1],k,coord[(j-1)*16+2*k]);
     printf("\n");
     } /* end j */

     count1 = count1 +1;
     } /* end while */

     if((totelems-count1*100) != 0) {
     fread((float *)coord+1,sizeof(float),16*(totelems-count1*100),
                                                        coordfile);

     for (j=1;j<=(totelems -count1*100);++j) {
     help =( (count1*100+(j-1)*12)+2L)*2L;
     fseek(connfile,help,0);
     fread((unsigned short *)&nrsubelem,sizeof(unsigned short),1,
                                                       connfile);
     printf("subelem : %4u \n",nrsubelem);
     for (k=1;k<=8;++k)
     printf("                 x%1u : %10f y%1u : %10f \n",k,
     coord[(j-1)*16+2*k-1],k,coord[(j-1)*16+2*k]);
     printf("\n");
```

```
     } /* end j */

            } /* end if */
            curlocat(24,Ø);
            printf("Hit any key to continue ...");
            getkey(&j);
            break;

case 2 :    do {
    printf("Enter the number of subelement : ");
    clearkbd();
    getint(4,1,-1,1,2,&i,Ø,1,9999);
    if (i>totelems)  printf("\n inputnumber is too large ! ");
    printf("\n");
    } while(i>totelems);

    printf("\n");
    fseek(coordfile,(loc[i]-1)*64L,Ø);
    fread((float *)coord+1,sizeof(float),16,coordfile);
    for (k=1;k<=8;++k)
    printf("                    x%lu : %1Øf y%lu : %1Øf \n",k,
                                  coord[2*k-1],k,coord[2*k]);
    printf("\n");
    .
    curlocat(24,Ø);
    printf("Hit any key to continue ..");
    getkey(&j);
    break;
case 3  :   exitflag = Ø;
default :   break;

   } /* end switch */


} while (exitflag) ;


fclose(locfile);
fclose(coordfile);
fclose(connfile);
free(connec);
free(coord);
free(loc);
}
```

(8.8) meshgraf.c

```
#include <stdio.h>
#include <malloc.h>
static char s[15],s_conn[15],s_coord[15];
static char *name = s,*nameconn=s_conn,*namecoord=s_coord;
static float *coord;

meshgraf()
{
FILE *connfile,*coordfile;
unsigned short totelems,max_gl_nodenr;
unsigned short countl=0,k,l,j;
int i,x[8],y[8],factor,factor_x,factor_y;
float min_x=1000.,min_y=1000.,max_x=-1000.,max_y=-1000.,help;

coord = (float *)calloc(1601,sizeof(float));
if(coord==NULL) {
   printf("*calloc() fails in meshgraf !");
   return;
   }

    clrscrn();
    printf("This subroutine does create a graphic display");
    printf("  of the mesh      \n");
    printf("generated by the meshgenerator !         \n\n");
    printf("Enter the name of the meshgenerator input filename\n");
    printf(" ( file extension '.key' )  :  ");
labell :     scanf("%s",s);
       i=0;
while (s[i] != '\0') {
      if (s[i] != '.' ) { i=i+1;continue;}
      break;
      }
if((s[i+1] != 'k')||(s[i+2]!='e')||(s[i+3]!='y')) {
   printf("The inputfile has a non valid file extension\n");
   printf("Try again : ");
   goto labell;
          }

for (j=0;j<=i;++j) {
    s_conn[j]  = s[j];
    s_coord[j] = s[j];
    }
    s_conn[i+1] = 'c';
    s_conn[i+2] = 'o';
    s_conn[i+3] = 'n';
    s_conn[i+4] ='\0';
   s_coord[i+1] = 'c';
   s_coord[i+2] = 'o';
   s_coord[i+3] = 'o';
   s_coord[i+4] ='\0';

if ( filexist(nameconn) == 0 || filexist(namecoord) == 0) {
   printf("\nEither    %s        \n",s_conn);
```

```
   printf("or        %s        \n",s_coord);
   printf("does not exist !                        \n");
   printf("Try again : ");
   goto label1;
   }

connfile = fopen(nameconn,"r+b");
coordfile = fopen(namecoord,"r+b");
fseek(connfile,2L,0);
fread((unsigned short *)&totelems,sizeof(unsigned short),1,
                                                    connfile);
 k = 8*totelems +1;
while((totelems - count1*100) >= 100 ) {
     fread((float *)coord+1,sizeof(float),1600,coordfile);
     for(k=1;k<=800;++k) {
   if ( min_x >= coord[2*k-1] ) min_x = coord[2*k-1];
   if ( min_y >= coord[2*k]   ) min_y = coord[2*k];
   if ( max_x <= coord[2*k-1] ) max_x = coord[2*k-1];
   if ( max_y <= coord[2*k]   ) max_y = coord[2*k];
        }
   count1 = count1 +1;
}

if((totelems - 100*count1) !=0 ) {
  fread((float *)coord+1,sizeof(float),16*(totelems-100*count1),
                                                    coordfile);
   for(k=1;k<=8*(totelems-100*count1);++k) {
   if ( min_x >= coord[2*k-1] ) min_x = coord[2*k-1];
   if ( min_y >= coord[2*k]   ) min_y = coord[2*k];
   if ( max_x <= coord[2*k-1] ) max_x = coord[2*k-1];
   if ( max_y <= coord[2*k]   ) max_y = coord[2*k];
        }
}

factor_x = 303.0/(max_x-min_x);
factor_y = 183.0/(max_y-min_y);
if( factor_x >= factor_y) factor = factor_y;
else                      factor = factor_x;
count1 = 0;
grafinit(2,0,0);
fseek(coordfile,0L,0);
while((totelems-100*count1) >= 100 ){
     fread((float *)coord+1,sizeof(float),1600,coordfile);
    for (k=1;k<=100;++k) {
    for (l=1;l<=8;++l) {
     x[l-1] = 8 + factor *( coord[ (k-1)*16 + 2*l-1 ] - min_x );
     y[l-1] = 191 - factor * (coord[ (k-1)*16+2*l ] - min_y);
    }
    grpoly(x,y,8,1);
    }
    count1 = count1 + 1;
} /* end while */
if((totelems-100*count1) != 0 ){
     fread((float *)coord+1,sizeof(float),16*(totelems-100*count1),
                                                    coordfile);
```

```
    for (k=1;k<=totelems-100*count1;++k) {
    for (l=1;l<=8;++l) {
        x[l-1] = 8 + factor *( coord[ (k-1)*16 + 2*l-1 ] - min_x );
        y[l-1] = 191 - factor * (coord[ (k-1)*16+2*l ] - min_y);
    }
    grpoly(x,y,8,1);
    }
} /* end if*/
grxlab("Hit any key to continue ..",7,54,0);
i = getkey(&j);
grafinit(0,0,0);
fclose(connfile);
fclose(coordfile);

free(coord);
}
```

DISTRIBUTION LIST

PROFESSOR G. CANTIN
Code 69Ci
DEPARTMENT OF MECHANICAL ENGINEERING
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

PROFESSOR M. HARTMANN (5)
Code 53Hw
DEPARTMENT OF MATHEMATICS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

PROFESSOR K. HETTLING
DEPARTMENT OF MATHEMATICS
MATHEMATICS/ASTRONOMY BUILDING
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VA 22903-3199

PROFESSOR R. NEWTON
Code 69Ne
DEPARTMENT OF MECHANICAL ENGINEERING
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

PROFESSOR Y. SHIN
Code 69Sg
DEPARTMENT OF MECHANICAL ENGINEERING
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

PROFESSOR L. WILLIAMSON
Code 53Wj
DEPARTMENT OF MATHEMATICS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DEPARTMENT OF MATHEMATICS
Code 53
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DEPARTMENT OF MECHANICAL ENGINEERING
Code 69
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

LIBRARY, Code 0142 (2)
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

RESEARCH ADMINISTRATION OFFICE
Code 012
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DUDLEY KNOX LIBRARY
3 2768 00335490 3